# Factorization-based Sparse Solvers and Preconditioners

**X. Sherry Li**
xsli@lbl.gov
Lawrence Berkeley National Laboratory

**Ichitaro Yamazaki, Esmond Ng**, LBNL
**Meiyue Shao**, Fudan University, China

SciDAC 2009, June 17, 2009, San Diego

## Outline

- Hybrid solver based on domain decomposition
  - direct + iterative
- Incomplete LU factorization preconditioner
  - Modify SuperLU, new dropping heuristics

- Funded through three SciDAC programs
  - TOPS

    Towards Optimal Petascale Simulations
  - CEMM

    Center for Extended MHD Modeling
  - ComPASS

    Community Petascale Project for Accelerator Science and Simulation

## Motivation

- Many large-scale numerical modeling codes require solution of sparse linear and eigen systems

  This talk →
  - Extended MHD equations in fusion plasma modeling
  - Maxwell equations in accelerator structure design
- Parallelism favors iterative solvers, ill-conditioning and indefinitness favors direct solvers
  - Iterative solvers scale well (mainly matrix-vector multiplication), but may suffer from slow convergence, and require robust preconditioners
  - Direct solvers need more memory, even more so for flops, less scalable because of high degree of task/data dependency
- Bridging the gap: use direct solver techniques as much as possible in the internal part of the iterative solver

## Hybrid solver

- Schur complement method

  = iterative substructuring method

  = non-overlapping domain decomposition

## Algebraic view

1. **Reorder into 2x2 block system, $A_{11}$ is block diagonal**

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

2. **Form the Schur complement**

$$S = A_{22} - A_{21} A_{11}^{-1} A_{12} = A_{22} - (U_{11}^{-T} A_{21}^{T})^{T} (L_{11}^{-1} A_{12})$$

$$\text{where} \quad A_{11} = L_{11} U_{11}$$

**S = interface (separator) variables**

3. **Compute the solution**

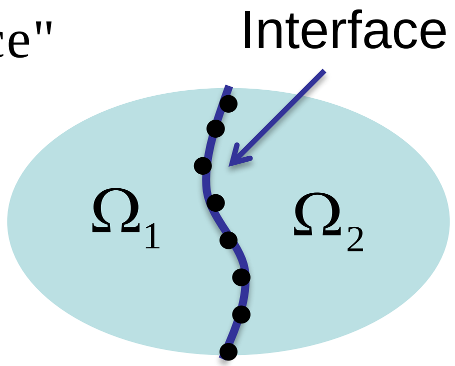
$$x_2 = S^{-1}(b_2 - A_{21} A_{11}^{-1} b_1)$$

$$x_1 = A_{11}^{-1}(b_1 - A_{12} x_2)$$

## Structural analysis view

- **Two subdomain case**

Substructure contribution: $A^{(i)} = \begin{pmatrix} A_{i\,i}^{(i)} & A_{i\,I}^{(i)} \\ A_{I\,i}^{(i)} & A_{I\,I}^{(i)} \end{pmatrix}$ $\quad i = \text{"interior"}$ $\quad I = \text{"Interface"}$



1. Assembled block matrix $A = \begin{pmatrix} A_{i\,i}^{(1)} & & A_{i\,I}^{(1)} \\ & A_{i\,i}^{(2)} & A_{i\,I}^{(2)} \\ A_{I\,i}^{(1)} & A_{I\,i}^{(2)} & A_{I\,I}^{(1)} + A_{I\,I}^{(2)} \end{pmatrix}$

2. Perform direct elimination of $A^{(1)}$ and $A^{(2)}$ independently,

   Local Schur complements: $S^{(i)} = A_{I\,I}^{(i)} - A_{I\,i}^{(i)} A_{i\,i}^{(i)^{-1}} A_{i\,I}^{(i)}$

   Assembled Schur complement $S = S^{(1)} + S^{(2)}$

## Solving the Schur complement system

- **Proposition** [Smith/Bjorstad/Gropp'96]

  For an SPD matrix, condition number of a Schur complement is no larger than that of the original matrix.

- **S is much reduced in size, better conditioned, but denser**
  - **solvable with preconditioned iterative solver**

- **Two approaches**
  1. **Explicit S (e.g., HIPS** [Henon/Saad'08], **and ours)**
     - **can construct general algebraic preconditioner, e.g. ILU(S), must preserve sparsity of S**
  2. **Implicit S (e.g.** [Giraud/Haidary/Pralet'09]**)** $S = S^{(1)} \oplus S^{(2)} \oplus S^{(3)} \ldots$
     - **preconditioner construction is restricted**
     - **E.g., additive Schwarz preconditioner**

$$M = {S^{(1)}}^{-1} \oplus {S^{(2)}}^{-1} \oplus {S^{(3)}}^{-1} \ldots$$

## Parallelism – multilevel partitioning

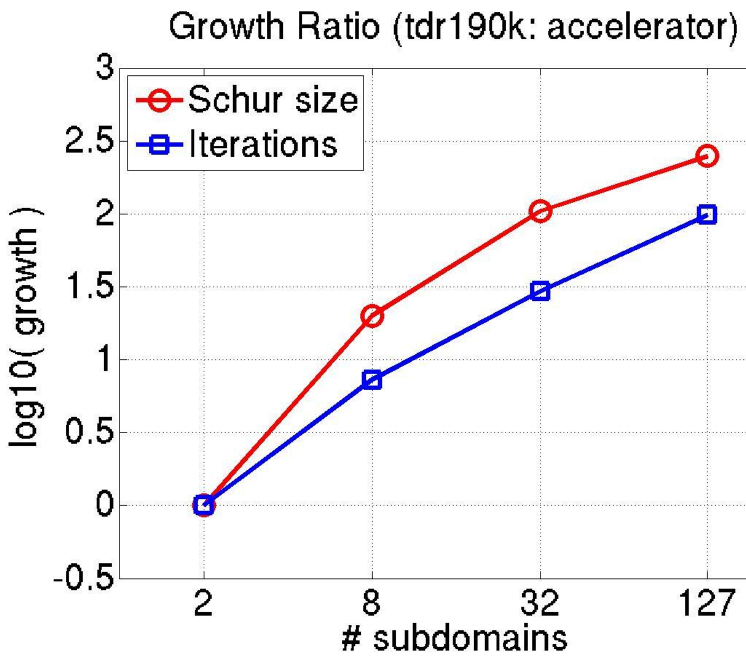
- **Nested dissection, graph partitioning**

$$\left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right) = \left(\begin{array}{cccc|c} A_{11}^{(1)} & & & & A_{12}^{(1)} \\ & A_{11}^{(2)} & & & A_{12}^{(2)} \\ & & \ddots & & \\ & & & A_{11}^{(k)} & A_{12}^{(k)} \\ \hline A_{12}^{(1)} & A_{12}^{(2)} & & A_{12}^{(k)} & A_{22} \end{array}\right)$$

- **Memory requirement: fill is restricted within**
  - **“small” diagonal blocks of $A_{11}$, and**
  - **ILU(S), sparsity can be enforced**
- **Two levels of parallelism: can use many processors**
  - **multiple processors for each subdomain direct solution**
    - **only need modest level of parallelism from direct solver**
  - **multiple processors for interface iterative solution**

## Previous work (recent): HIPS – Hybrid Iterative Parallel Solver [*P. Henon* and *Y. Saad*, 2008]

- Based on Hierarchical Interface Decomposition
- Major limitation: number of processors = number of subdomains
- Dilemma: large number of subdomains needed for parallelism, small number of subdomains needed for convergence



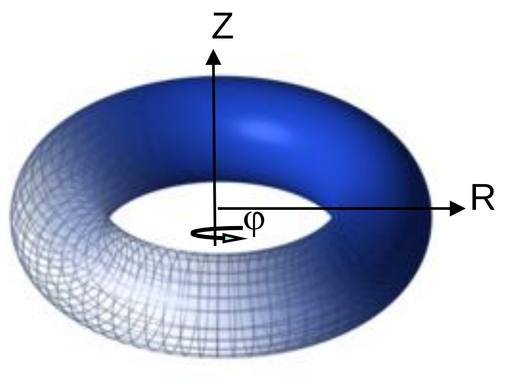
## Our implementation

- Flexibility and robustness
  - multiple processors to solve each subdomain
  - subset of processors to solve the Schur complement system
  - less restriction on sparsity pattern of ILU(S)
- High performance
  - usage and extension of state-of-art software

    ParMETIS, PT-SCOTCH, SuperLU, SuperLU_DIST, PETSc
  - scalable computation of Schur complement
    - parallel symbolic computation to set up data structure
    - exploit dense blocks (supernodes) to improve efficiency
  - ILU of sparsified Schur complement: $\sigma_1, \sigma_2$ to control dropping
    - parallel symbolic factorization for time efficiency
    - no duplicate data for memory efficiency

## Application 1: Fusion plasma study

- SciDAC Center for Extended Magnetohydrodynamic Modeling (CEMM), PI: S. Jardin, PPPL
- Develop simulation codes for studying the nonlinear macroscopic dynamics of MHD-like phenomena in magnetized fusion plasmas, and address critical issues facing burning plasma experiments such as ITER
- Simulation code suite includes M3D-$C^1$, NIMROD



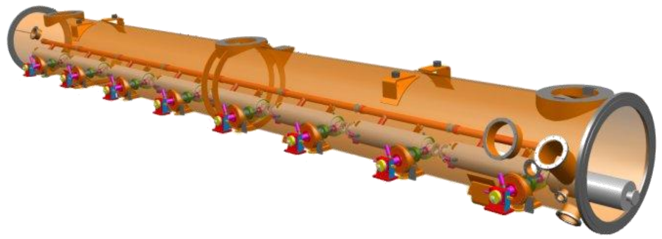
[S. Jardin]

- At each $\varphi$ = constant plane, scalar 2D data is represented using 18 degree of freedom quintic triangular finite elements $Q_{18}$
- Coupling along toroidal direction

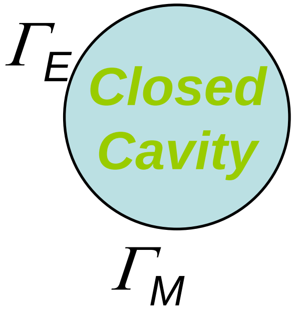
## Application 2: Accelerator cavity design

- Community Petascale Project for Accelerator Science and Simulation (ComPASS), PI: P. Spentzouris, Fermilab
- Development of a comprehensive computational infrastructure for accelerator modeling and optimization
- RF cavity: Maxwell equations in electromagnetic field
- FEM in frequency domain leads to large sparse eigenvalue problem; needs to solve shifted linear systems

[L.-Q. Lee]

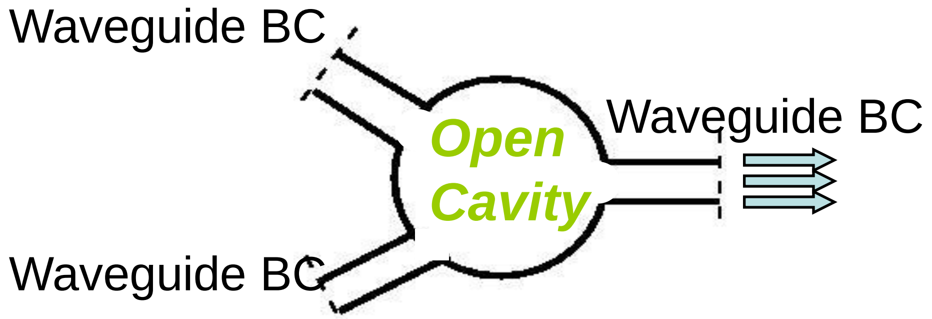


RF unit in ILC



linear eigenvalue problem

$$(K_0 - \sigma^2 M_0)\, x = M_0\, b$$

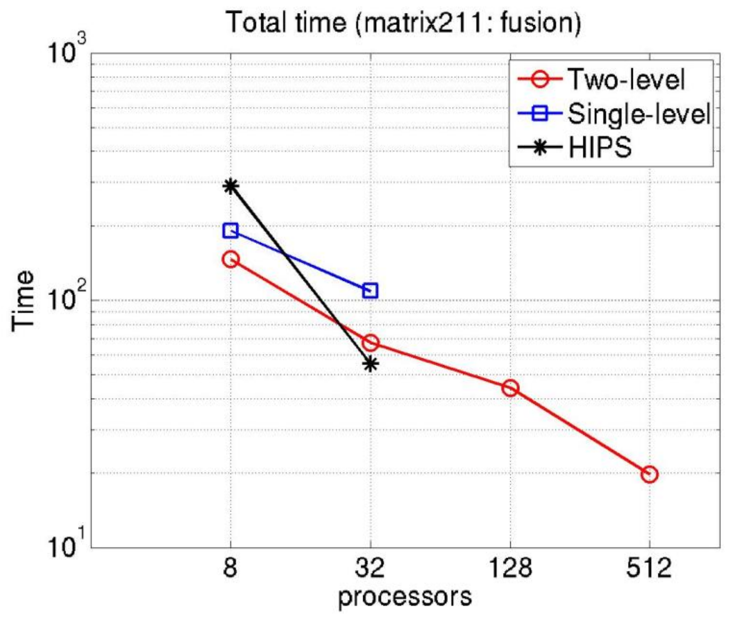


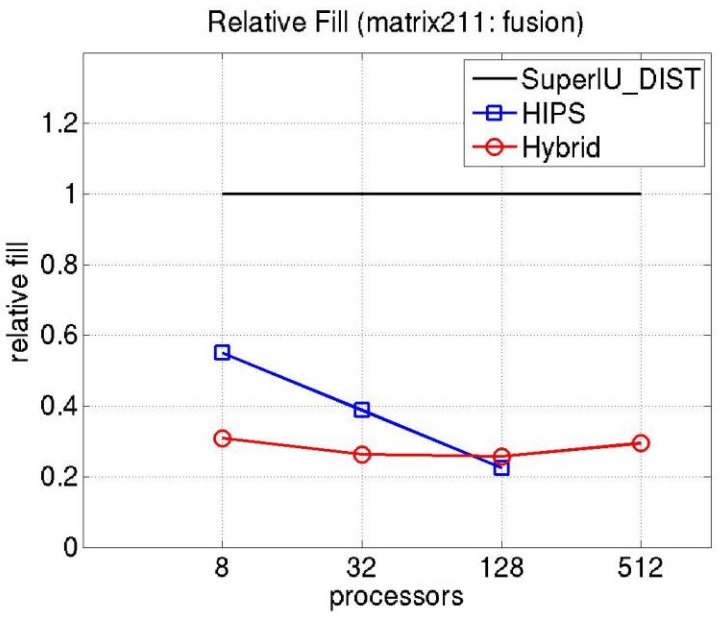
nonlinear complex eigenvalue problem

$$(K_0 + i\,\sigma W - \sigma^2 M_0)\, x = b$$

## Hybrid solver result: matrix211

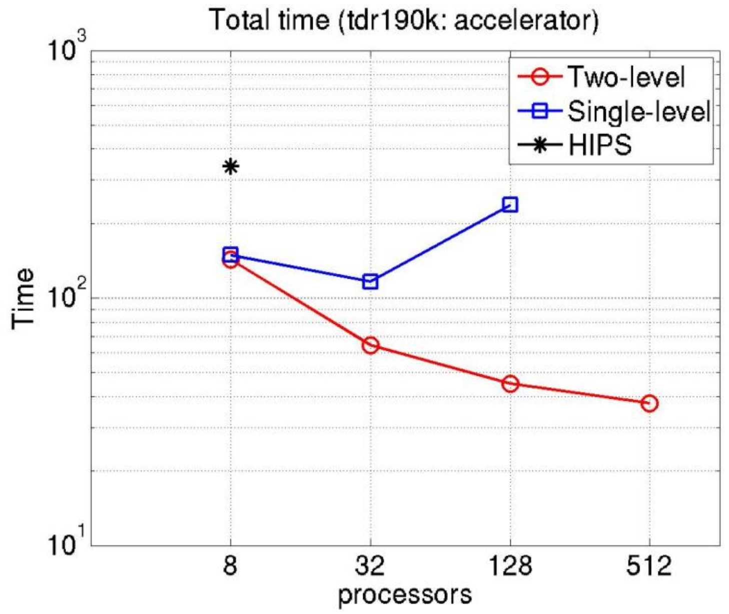
- **Fusion M3D-$C^1$ code, extended MHD modeling (PPPL)**
- **dimension = 801,378**
- **drop tolerance: $\sigma_1=10^{-5}$ to preserve sparsity of S**
- **Two-level: fixed 8 subdomains**
- **Strong scaling: time & memory**

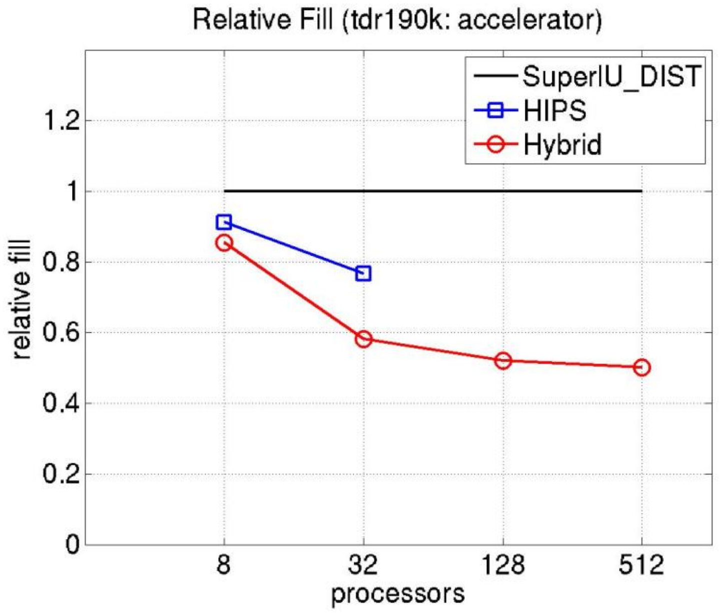




## Hybrid solver result: tdr190k

- **Modeling particle accelerator (SLAC)**
- **dimension = 1,100,242**
- **drop tolerance: $\sigma_1=10^{-5}$ to enforce sparsity of S**
  - **about half of the nonzeros are discarded**
- **Strong scaling: time & memory**





## Hybrid solver summary

- Our hybrid solver achieves convergence within 30 iterations
  - Convergence “independent” of #procs
- Flexible in exploiting parallelism, more robust than HIPS
  - Using 32 processors and beyond, HIPS does not converge within 1,000 unrestarted GMRES iterations
- Scales better than SuperLU_DIST,  needs much less memory

- Future work
  - Compare with “implicit Schur” approach
  - Larger problems, larger processor count

## Incomplete LU factorization preconditioner

- Modify SuperLU, new dropping heuristics

## ILU preconditioner

- **A very simplified view:**

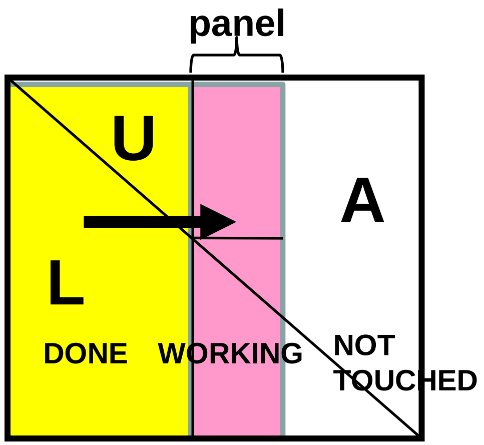
  $A = \tilde{L}\tilde{U} + E, \;\; \text{if } \| E \| \text{ small}, \;\; (\tilde{L}\tilde{U})^{-1} A \;\text{ may be well conditioned}$

  $\text{Then, solve } (\tilde{L}\tilde{U})^{-1} Ax = (\tilde{L}\tilde{U})^{-1} b \;\text{ iteratively}$
- **Structure-based dropping: level of fill**
  - **ILU(0), ILU(k)**
  - **Rationale: the higher the level, the smaller the entries**
  - **Separate symbolic factorization step to determine fill-in pattern**
- **Value-based fropping: drop truly small entries**
  - **Fill-in pattern must be determined on-the-fly**
- **ILUTP [Saad]: among the most sophisticated, and (arguably) robust**
  - **“T” = threshold, “P” = pivoting**
  - **Implementation akin to direct solver**
- **We use SuperLU code base to perform ILUTP**

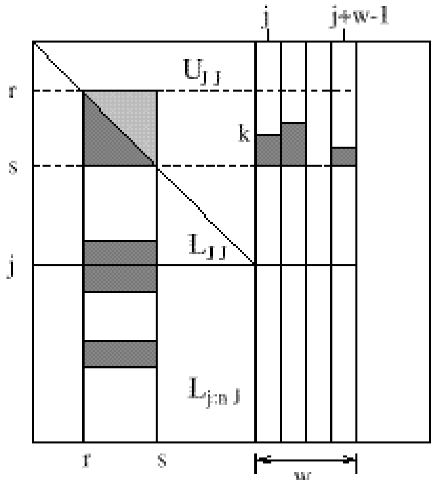
## SuperLU [Demmel/Eisenstat/Gilbert/Liu/Li ’99]

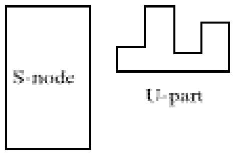
http://crd.lbl.gov/~xiaoye/SuperLU

- **Left-looking, supernode**



1. Sparsity ordering of columns use graph of A’*A
2. Factorization
   For each panel …
   - Partial pivoting
   - Symbolic fact.
   - Num. fact. (BLAS 2.5)
3. Triangular solve





## Primary dropping rule: S-ILU(tau)

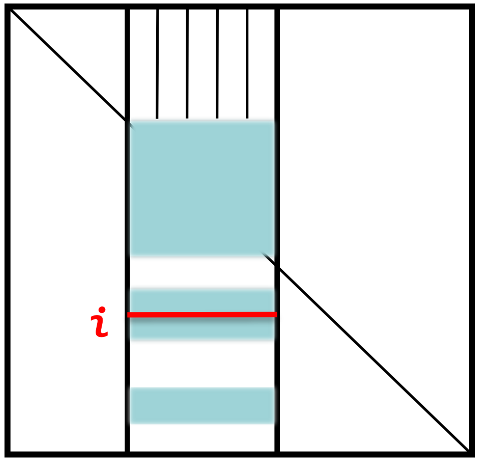
- **Similar to ILUTP, adapted to supernode**
  1. **U-part:**

     $$\text{If } |u_{ij}| < \tau \cdot \|A(:,j)\|_\infty, \text{ then set } u_{ij} = 0$$

  2. **L-part: retain supernode**

     $$\text{Supernode } L(:,s:t), \text{if } \|\mathrm{L}(i,s:t)\|_\infty < \tau, \text{then set the entire } i\text{-th row to zero}$$

- **Compare with scalar ILU(tau)**
  - **For 54 matrices, S-ILU+GMRES converged with 47 cases, versus 43 with scalar ILU+GMRES**
  - **S-ILU +GMRES is 2.3x faster than scalar ILU+GMRES**

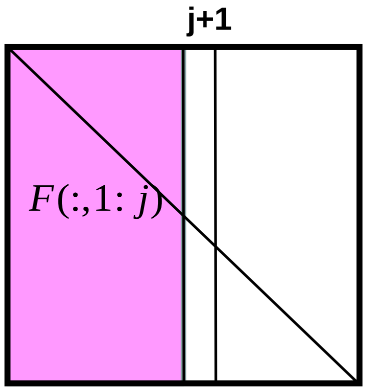


## Secondary dropping rule:  S-ILU(tau,p)

- **Control fill ratio with a user-desired upper bound** $\gamma$
- **Earlier work, column-based**
  - **[Saad]: ILU(tau, p), at most p largest nonzeros allowed in each row**
  - **[Gupta/George]: p adaptive for each column** $p(j) = \gamma \cdot nnz(A(:,j))$
    **May use interpolation to compute a threshold function, no sorting**

- **Our new scheme is "area-based"**
  - Look at fill ratio from colum 1 up to j:
    $$fr(j) = nnz(F(:,1:j)) / nnz(A(:,1:j))$$
  - **Define adaptive upper bound function** $f(j) \in [1, \gamma]$
    If $fr(j)$ exceeds $f(j)$, retain only p largest, such that $fr(j) \le f(j)$

➢ **More flexible, allow some columns to fill more, but limit overall**



## Experiments: GMRES + ILU

- **Use restarted GMRES with our ILU as a right preconditioner**

  Solve $PA(\tilde{L}\tilde{U})^{-1}y = Pb$

- **Size of Krylov subspace set to 50**

- **Stopping criteria:** $\|b\text{-}Ax_k\|_2 \le 10^{-8}\|b\|_2$ and $\le 1000$ iterations

## S-ILU for extended MHD calculation (fusion)

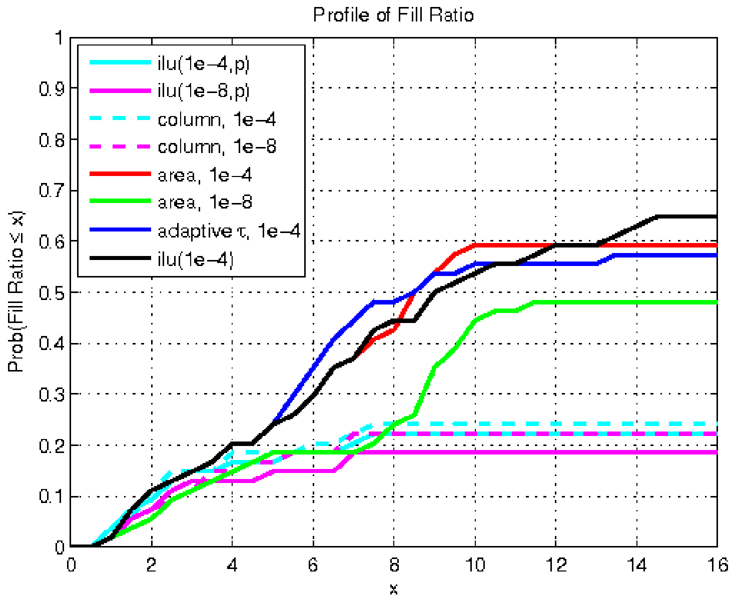
- **Opteron 2.2 GHz (jacquard at NERSC), one processor**
- **ILU parameters: drop_tol = 1e-4, gamma = 10**

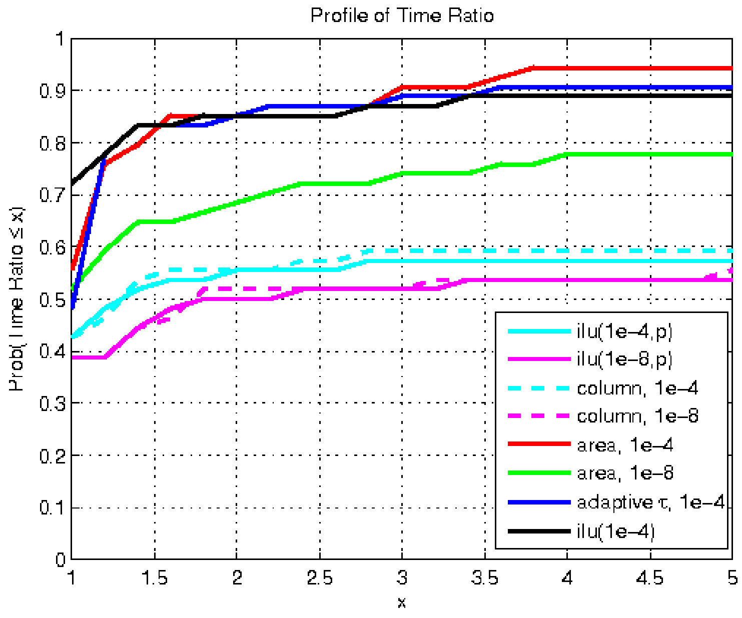
- **Up to 9x smaller fill ratio, and 10x faster**

| Problems | order | Nonzeros (millions) | ILU | | GMRES | | SuperLU | |
|---|---|---|---|---|---|---|---|---|
| | | | time | fill-ratio | time | iters | time | fill-ratio |
| matrix31 | 17,298 | 2.7 m | 8.2 | 2.7 | 0.6 | 9 | 33.3 | 13.1 |
| matrix41 | 30,258 | 4.7 m | 18.6 | 2.9 | 1.4 | 11 | 111.1 | 17.5 |
| matrix61 | 66,978 | 10.6 m | 54.3 | 3.0 | 7.3 | 20 | 612.5 | 26.3 |
| matrix121 | 263,538 | 42.5 m | 145.2 | 1.7 | 47.8 | 45 | fail | - |
| matrix181 | 589,698 | 95.2 m | 415.0 | 1.7 | 716.0 | 289 | fail | - |

## S-ILU comprehensive tests

- 54 matrices: Matrix Market, UF Sparse Matrix, fusion
- Performance profile of fill ratio – fraction of the problems a solver could solve within a fill ratio of X
- Performance profile of runtime – fraction of the problems a solver could solve within a multiple of X of the best solution time among all the solvers



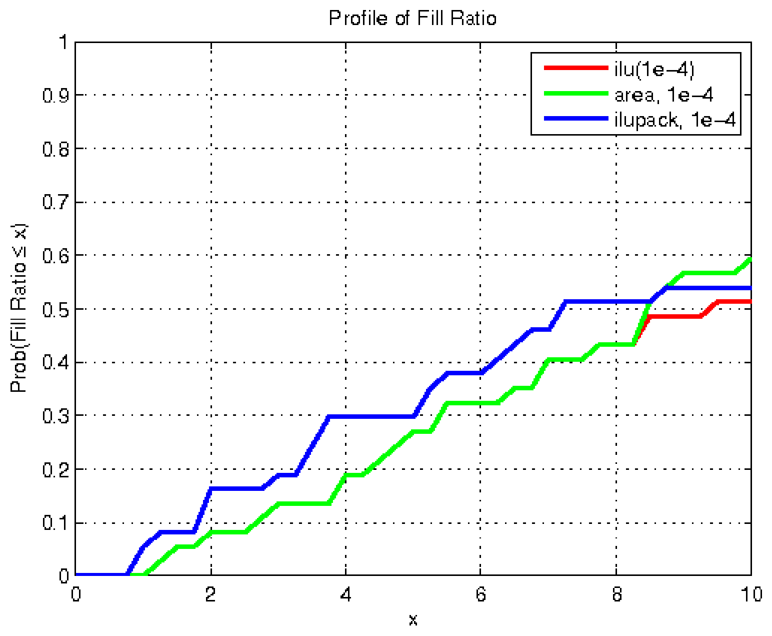


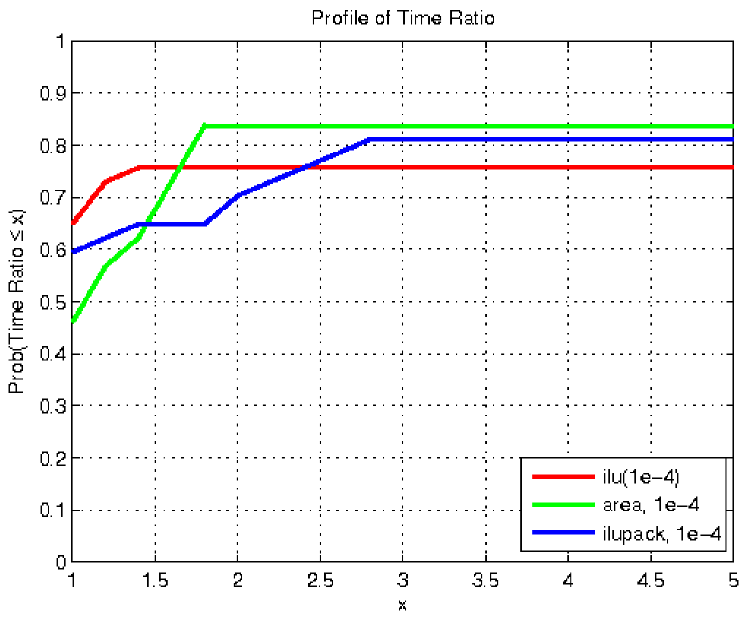
- **Conclusion:**
  - New area-based heuristic is much more robust than column-based one
  - ILU(tau) is reliable; but need secondary dropping to control memory

## Compare with ILUPACK [Bolhoefer et al.]

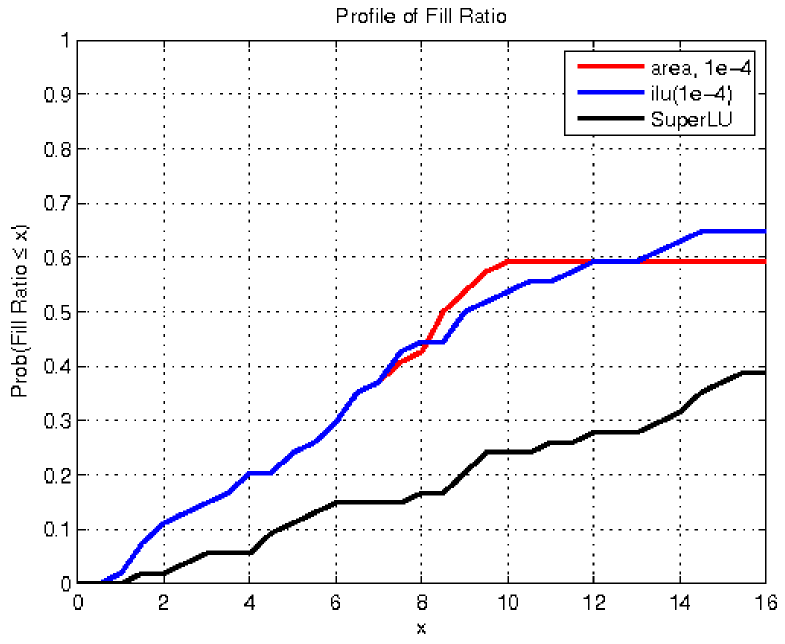
- **ILUPACK: inverse-based multilevel method**
- **Parameters:**
  - **S-ILU:** $\tau = 10^{-4}$, area-based, $\gamma = 10$, COLAMD ordering $G(A^T A)$
  - **ILUPACK:** $\tau = 10^{-4}, \nu = 5, \gamma = 10$, AMD ordering $G(A^T + A)$

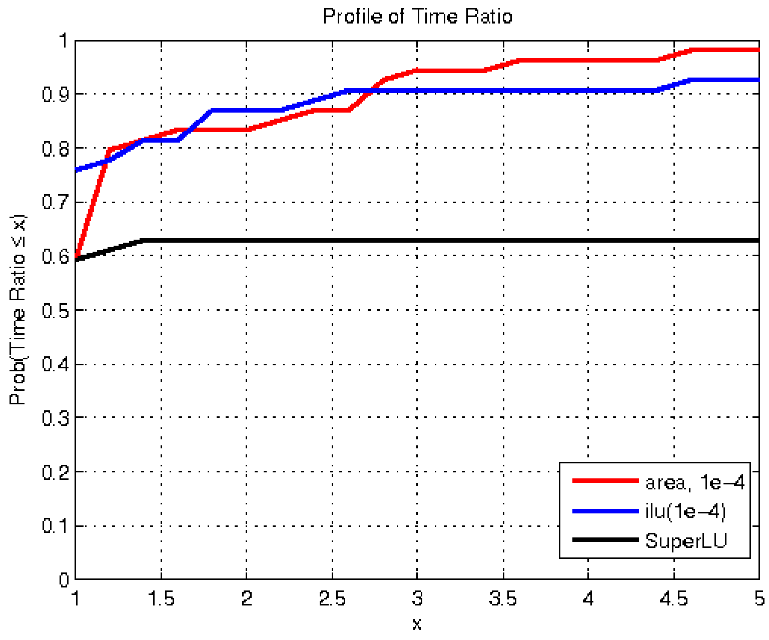
- **37 test matrices, one processor Xeon 2.5 GHz**





## Compare with direct solver SuperLU





- Works for over 60% of the comprehensive test problems (54)
  When it works, it is much faster than direct solver

## ILU summary

- New supernodal, area-based dropping is more reliable and faster than classical column-based ILUTP
  - Fusion matrices: 9x reduction in fill, 10x faster than LU
- Competitive with an inverse-based multilevel ILU method: ILUPACK
- Available in forthcoming release of SuperLU v 4.0
- Future work: parallel ILU in SuperLU_DIST

- **Sparse matrix factorization is very hard to scale up because of high degree of dependency, but moderate parallelism is achievable.   It can be effectively used to improve numerics for iterative methods.**